Stanford CS 229 Fall 2017

Final Project Report: Real Time Tennis Match Prediction Using Machine Learning

Yang "Eddie" Chen, Yubo Tian, Yi Zhong

December 15, 2017

Sports bring unpredictability and a lucrative betting industry trying to predict the unpredictable. While past work on predicting outcome for tennis matches focused on pre-game prediction using historical data, this project adopts an innovative data model by combining both historical performance data and real-time player in-game stats, and apply machine learning to predict tennis match outcome in-game - namely, after the 1st set and after the 2nd set. This project explores and compares four data models: (1) historical data only (pre-game prediction; baseline), (2) current after set 1 data only (in-game prediction), (3) historical + current after set 1 data (in-game prediction), and (4) historical + current after set 2 data (in-game prediction), with various machine learning techniques such as logistic regression, support vector classification (SVC) with linear, rbf and poly kernels, neural network, and Naive Bayes. Feature selection techniques such as recursive feature elimination, and principal component analysis were employed as well. W'e find that historical performance data do not increase accuracy when making in-game prediction with 1st set performance data, and in fact accuracy and precision both improved when in-game data are introduced. More importantly, a thorough analysis on results was done to reveal and understand the predominant challenge in this data model - high bias, as our collected feature set does not cover edge cases. **TTL** (total points won) is found to be the most dominant and predictive feature, with other standout features also revealed. From there, the project outlines the future work needed to combat the high bias issue by proposing other features that could be used but not included in the current project.

1 Introduction

In the 2017 Stuttgart Open, Roger Federer, then an 18-time grand slam champion, lost to the world No. 302 player Tommy Haas on a grass court, which hadn't happened since 2002. Based on past performance alone, almost any model would have predicted a Federer win in pre-game bets. This motivates us to apply machine learning to predict tennis matches in real time; in particular, we want to apply a hybrid model that combines historical and real-time, in-game information.

2 Related Work

Extensive research has been done on tennis as it is an ideal candidate for using hierarchical probability models: a match consists of a sequence of sets, which in turn consist of several games, which in turn consist of a sequence of points. Knottenbelt [5] proposed a common opponent model to extract features between two players that is found to be very predcitve in his stochastic models, while James [3] has done thorough study on probability Formulas and statistical analysis on tennis matches. Recent researchers began to apply machine learning on tennis pre-game predictions based on past performance (including previous encounter, current ranking, etc), such as Sipko [9], and Madurska [6] - whose by-set method inspired us as well.

Our project seeks to apply classification algorithms from machine learning to model men's professional singles matches by using both pre-game, historical performance calcualted via a common opponent model [5], and real-time, in-game stats [4] [6]. As pointed out by Sipko [9], this in-game approach "allows the model to capture the change in a player's performance over the course of the match. For example, different players fatigue during a match in different ways." We hope that results from our predictions can be extended to give real-time coaching advice to support game strategy decision.

3 Dataset and Features

3.1 Dataset & pre-processing

This project employs two open source datasets mainly for labels and features, by Jeff Sackmann [8] and [4]. The *Match Charting Project* contains set level data from 1442 matches, spanning from 1974 to 2017. *Tennis ATP* dataset contains match level data from 1969 to 2017.

- **Date**: Matches from 2000 - 2017 are used as samples (1415 matches)
- **Remove duplicates via match id**: Since both datasets are manually entered via crowdsourcing, there are some inaccuracy and duplication that require our pre-processing.
- **Remove Davis Cup**: Davis Cup is the World Cup of tennis, where players play for their country. We decided to remove all Davis Cup entries since group matches may include strategic moves that are considerably different from individual tournaments.
- **Merge Tennis ATP and Match Charting Project**: Real-time stats are obtained the *Match Charting Project* dataset, whereas historical performance for both players is computed from *Tennis ATP*. Two datasets are merged on player's first and last name, with players of the same names removed manually.
- **Feature Standardization** We are unable to extract all features listed in Sipko [9]. While extracting features from *Tennis ATP* was easy, the *Match Charting Project* data proves much more challenging. We managed to extract a sbset of all possible features from processing the point-by-point data in the *Match Charting Project*, and standardized between the historical and current performance.

3.2 Feature Extraction

Original dataset we used denotes Player 1 as the winner and Player 2 as the loser. While processing, we decide to represent each set in each match with two rows, one from each player's perspective, and label the match result accordingly. Thus we have one row per player per set per match, containing features from historical data and real time data. Historical data includes match details and player's past performance; real-time data, for the scope of this project, will be defined as the performance from all previous sets in the match:

1. A vector of input features ($\mathbf{X}$), describing the performance in previous set and historical performance from one player's perspective
2. A target value ($\mathbf{y}$), indicating the match outcome. $y \in$ '$\{1, -1\}$

3.2.1 Symmetric Feature Representation

There are two approaches to represent features from two players. First, we can have two features for the same metric (e.g. $RANK_1$ and $RANK_2$). This approach doubles the number of features, but also provides a quantitative measure that we can compare across all players. Second, we can look at *differences* between two players (e.g. $RANK_{DIFF} = RANK_1 - RANK_2$). This would reduce the number of features, but does not allow the model to compare across players. Previous research has shown that the difference variables are predictive enough [9] [3] for past performance (on match level). For current performance (on set level), we evaluated the performance of both approaches. There was no difference in terms of accuracy, hence the difference model was used for current match as well for consistency.

3.2.2 Common Opponent Model for Past Performance

We use a method proposed by Knottenbelt [5] and extended by Sipko [9] to construct *difference variables* capturing player characteristics and past performance, by using their **common opponents** to achieve a fair comparison. The detailed code can be found in our codebase, while the process works as follows:

- Identify a set of common opponents between the two players of a given match.
- Compute the average performance of both players against the common opponent.
- Take the difference between average performance for each extracted feature.

3.2.3 Edge Cases for Common Opponents

We have run into 2 edge cases: (1) when at least one of the players is new; (2) when two players do not have any common opponents. Upon investigation, our solution is: (1) when the player is new with no historical data, we remove the samples, which consist of 84 data points out of 1421 matches; (2) when two players do not have any common opponents, we compute the average for each player (regardless of opponents) and take the difference, which consist of 74 out of 1421 matches. Note, as described in Section 3.2, our final sample size will contain $2x * 1421$ rows, where x is the number of sets played in each match.

3.3 Feature Lists

3.3.1 Features from historical match performance

Features are the *difference variables* averaged for both players obtained from a common opponent model [5], unless otherwise noted. They are:

- **Performance Features**: Number of aces, double faults, break points faced and saved; Percentages of winning on first and second serves, and first serve in rate.

- **Match Details**: Match duration, number of serve games, number of serve points per game
- **Player Bio**: ATP Rank, ATP Rank Points, same handedness, height

3.3.2 Features from current match performance

Features are the difference in average for a snapshot in the match: after set 1, and after set 2. They are:

- **Performance Features** Number of aces, double faults, winners and total points won; Percentage of winning on first and second serve, return points won, first serve in
- **Match Details** Surface (hard, grass, or clay), match type
- **Player Bio** Same handedness

4 Method

4.1 Logistic Regression

As a start, we built a logistic regression on the data using Scikit-Learn's implementation (add citation). The $L2$-penalized model minimizes the cost function below:

$$\min_{w,c} \sum_{i=1}^{n} \log(e^{-y_i(X_i^T w+c)} + 1) + \frac{1}{2} w^T w$$

The solver implemented in the code uses a coordinate descent algorithm, and the detailed proof can be found in Hsieh 2008 [2]

4.2 Support Vector Machine

Next, we used a linear support vector classifier from scikit-learn [7] to predict the tennis match outcome. Given training vectors $x_i \in \mathbb{R}^n$ and the label vector $y = \{1, -1\}$ where 1 represents a win and -1 represents a loss, we can formulate the support vector machine as solving the following primal problem:

$$min_{w,b,\varsigma} \sum_{i=1}^{n} \varsigma_i + \frac{1}{2} w^T w$$

$$s.t. y_i(w^T \theta(x_i) + b) \geq 1 - \varsigma_i, \varsigma_i \geq 0$$

Its dual is

$$\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha$$
$$s.t. y^T \alpha = 0$$
$$0 \leq \alpha \leq 1$$

where e is the vector of all ones, and $Q_{i,j} = y_i y_j K(x_i, x_j)$, and $K(x_i, x_j)$ is the kernel here. We tried different kernels including linear, RBF, and poly which give similar performances.

4.3 Other algorithms

Moreover, we've implemented **Naive Bayes** as a classifier, to check the model performance. We were curious to test if Naive Bayes classifier works well; and if so, if it indicates that the features are *conditionally independent* - i.e. break point percentage, rank difference, and all other features don't really relate to each other as long as I win the matches. Given a class variable $y = \{1, -1\}$ and a dependent feature vector x_1 through x_n, BayesâĂŹ theorem states the following relationship:

$$P(y|x_1, ..., x_n) = \frac{P(y)P(x_1, ..., x_n|y)}{P(x_1, ..., x_n)}$$

In particular, we have chosen the Gaussian Naive Bayes algorithm following the assumption that X with respect to $y = \{1, -1\}$ is distributed according to a Gaussian distribution. This gives us the feature likelihood as

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2})$$

And we use the following classification rule:

$$\hat{y} = arg \max_y P(y) \Pi_{i=1}^{n} P(x_i|y)$$

We mainly used Naive Bayes to benchmark model performance against logistic regression and linear SVC, as it is extremely fast to run.

5 Experiments, Results, Discussion

We organized discussion around results and experiments in the following 4 questions. We mainly recorded results using logistic regression and linear SVC, for their excellent performances under our various data models. Moreover, cross-validation (10-fold) is done on all the results to ensure robustness. The general random split of train-dev-test (60:20:20) is also observed. Wr compared different models using the dev set after training models on the train set; once we had the best model, we passed that through the test set for final model evaluation.

5.1 Parameters, Metrics

For logistic regression, we chose L2 regularization with the *liblinear* Algorithm [2] to use in the optimization solver. For small datasets, *liblinear* is shown to perform really well. The penalty function makes sure that we don't overfit. For SVC, we went with linear kernel after exploring multiple options such as linear, poly and rbf based on empirical performance on dev set. We also used *squared hinge loss* ($\max(0, -)^2$) to penalize violated margins more strongly, based on emprical evidence on our particular dataset. Moreover, we used L2 regularization as well, as penalizing large weights tends to improve generalization (because it means that no input dimension can have a very large influence on the scores all by itself) and thus avoids overfitting. As evident by results presented below, this is part of our effort to encourage models to prefer smaller and more diffuse weight vectors (to counteract the fact that a very small subset of features account for most predictive power). The classifier is thus "encouraged" to take into account all input dimensions to small amounts rather than a few input dimensions and very strongly, ideally.

We mainly looked at **accuracy** as our success criterion, which calculates how many labels our model predicts correctly out of all the dev/test labels.

5.2 How well can we predict tennis matches?

Our best model, which uses both historical and ingame performance with SVC predicts with 88% accuracy on dev set when 2 sets have been played. The detailed performance comparison of each data method (with best performing machine learning algorithm) can be seen in Table 5.1

Table 5.1: Different Modeling Method Test Accuracy Comparison

Data Model	**Algorithm**	**Accuracy**
Historical Data Only	SVC with rbf kernel	83%
Real-Time Data Only, After Set 1	Logistic Regression	81%
Historical + Current Data (after Set 1)	Linear Linear SVC	79%
Historical + Current Data (after Set 2)	Linear SVC	86%

5.3 Does using current, real-time info improve accuracy?

It does. Moreover, we found that if we are using current, real-time info, there is really no need for past performance data. The model that utilizes real-time data after the first set only showed accuracy on par with that plus historical data. This surprising result arises despite the fact the features from current match are fewer than historical data, which have been researched extensively. We have shown the following confusion matrices to document each data model's performance in Tables 5.2, 5.3, 5.4, 5.5. Moreover, the accuracy graph also shows how using real-time info improves accuracy in Figure 5.1.

Interestingly, models built on historical data only show a much greater variance when it comes to performance (we have observed accuracy from 83% to 92%), while models using current data are much more stable, giving accuracies in the range of ±1%. Here, we compare performance based on average values obtained from running models.

Figure 5.1: Accuracy vs. Number of Features

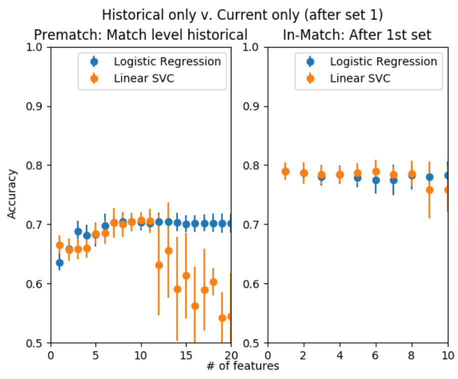


Table 5.2: Historical Only

	TRUE	**FALSE**
POSITIVE	4	5
NEGATIVE	7	8

Table 5.3: Real-Time Data After Set 1

	TRUE	**FALSE**
POSITIVE	4	5
NEGATIVE	7	8

5.4 What features are the most predictive?

TTL (total points won), which indicates how many points one player has won in the match up till our cut-off point (either after Set 1 or 2, depending on the run), is the most important feature by miles: This result is computed using recursive feature elimination across our data models and different prediction model. Adding more features only leads to minimal improvement at best. In most case, having **TTL** alone almost always yields the best performance. Other top features include: **RCV** (% of return points won) from real-time data, **BPP** (breaking point saving %) if only historical performance data are used as features. Because of suspected collinearity and overfittng, we wanted to make sure we don't consider all features, but really understand how performance changes as we go from the most predictive feature to the least predictive feature. Since we are using logistic regression and linear SVC mainly, we looked at recursive feature elimination to find the optimal number of features (by recursively considering smaller and smaller sets of features). The graphs of how accuracy changes with number of features are included below in Figure 5.2; notice how it drops over time due to L2 regularization present in both algorithms.

Table 5.4: Historical + After Set 1

	TRUE	FALSE
POSITIVE	4	5
NEGATIVE	7	8

Table 5.5: Historical + After Set 2

	TRUE	FALSE
POSITIVE	4	5
NEGATIVE	7	8

5.5 Diagnostics

We observed the problem of **high bias** (high training error; Small gap between training and test error) after plotting the learning curves in Figures 5.3 and 5.4.

To combat high bias, we turn to unimplemented new features as this is the best solution outlined in Andrew Ng's error analysis slides. Despite other available features, our models learned that the information contained in other additional features (such as number of double faults or aces, etc.) is mostly already captured by **TTL**. Therefore, we need to find features that are complement to **TTL**, revealing information that could let to a "come back" even if the player is at the moment behind, in order to get a more accurate prediction.

Two features that we added after the poster session for this purpose were:

1. **duration win/loss rate**: calculated as the average difference in duration (in minutes) of the past matches each player won or lost respectively, using the Common Opponent Model. When predicting the result of a new match, we divide the difference by the elapsed time of the match that we are trying to predict.

2. **age diff win/loss rate**: For this feature, we calculate the average age difference of the past matches each player won or lost respectively, using the Common Opponent Model. Then, when predicting a new match, we divide the difference by the age difference between the two players in the match that we are trying to predict.

Figure 5.2: Accuracy vs. Number of Features

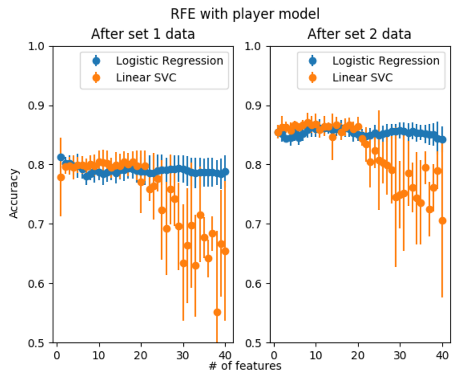


Figure 5.3: Logistic Regression Learning Curve

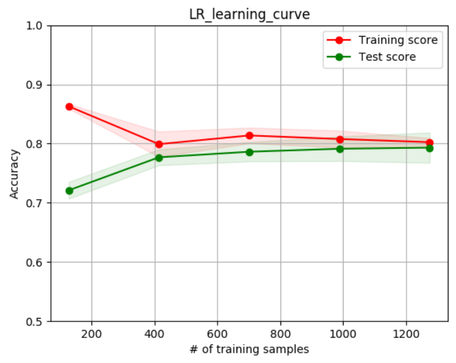
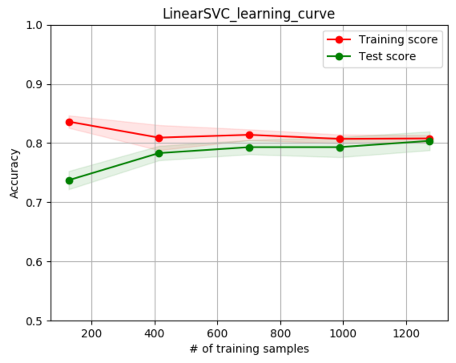


Figure 5.4: Linear SVC Learning Curve



However, after adding these two features, we realized on the historical data only data model benefited, with better accuracy achieved when using SVC with a rbf kernel.

6 Future Work

For future work, there are still unimplemented features that we think will improve the model accuracy, such as: the (relative) importance of tournament/match for the player, number of matches player shad in the days before this match (as a proxy for fatigue), the tightness in schedule/intensity of those matches, number of matches the player played before in similar environment (location, weather, surface), etc.

7 Conclusion

The use of common opponent model and difference variables already offer high predictive power in simple machine learning models such as logistic regression and Naive Bayes, and it's insightful to learn that for real-time prediction, total points won from before can really tell who is more likely to win, and a player's performance when faced with break points directly indicates his chance of winning. We also learnt that when making in-game prediction with 1st set performance data, historical performance data do not increase accuracy - i.e. most players actually do get "fresh starts". Moreover, when in-game data are introduced, accuracy and precision both improved. We also showed that high bias is the main issue for tennis prediction models, and any future model should take this into consideration.

8 Contributions

8.1 Yang "Eddie" Chen

- Researched, reviewed and synthesized relevant literature for applicable ideas and the Related Work section, such as Clarke and Dyte [1] and O'Malley [3] for difference variable, and Knottenbelt [5] for the common opponent model
- Implemented the Common Opponent Model to look up players stats from common opponents
- Generated ATP Match results for labels
- Applied math fomulations from machine learning models & techniques learned from class, namely logistic regression, SVM and Naive Bayes.
- Conducted error analyses along with Yi & Yubo to understand the root cause of our performance curve, and proposing new checks and solutions.
- Typsetted (LaTeX) the project milestone report, contributed mostly in the poster making, and typesetted & synthesized the final report.

8.2 Yubo Tian

- Feature extraction: generated real-time set-level performance data from Sackmann's MCP [4]
- Data labeling: extracted set level match results from player points information from Sackmann's MCP [4]
- Data processing: joined match-general information, historical data and real-time match data; fixed various issues encountered during the process; wrote scripts to generate all data and labels upon one click
- Initial modeling/error analysis: trained logistic regression model on joined data
- Feature analysis and improvement: after initial modeling, analyzed feature selection results and train/dev errors, designed and extracted additional features in an attempt to fix the high bias problem exposed
- Plot and Report: wrote scripts to plot learning curves; contributed to milestone/ final report contents.

8.3 Yi Zhong

- Feature extraction: get match details related features from MatchChartingProject,
- Data processing: Joined data by linking real-time match data from MCP [4] to match details data from ATPMatches [8] (methods: playerID, playerName etc) (with Yubo).
- Modeling: Wrote the script for logistic, SVC and other model exploration for different data model
- Error analysis: Wrote the graph script for recursive feature elimination, confusion matrix, feature ranking

References

[1] Stephen R. Clarke and David Dyte. Using official ratings to simulate major tennis tournaments. *International Transactions in Operational Research*, 7(6):585 – 594, 2010.

[2] Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S Sathiya Keerthi, and Sellamanickam Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th international conference on Machine learning*, pages 408–415. ACM, 2008.

[3] O'Malley A. James. Probability formulas and statistical analysis in tennis. *Journal of Quantitative Analysis in Sports*, 4(2):1–23, April 2008.

[4] et al. Jeff Sackmann. The tennis abstract match charting project, 2017.

[5] William J. Knottenbelt, Demetris Spanias, and Agnieszka M. Madurska. A common-opponent stochastic model for predicting the outcome of professional tennis matches. *Computers and Mathematics with Applications*, 64(12):3820 – 3827, 2012. Theory and Practice of Stochastic Modeling.

[6] Agnieszka M. Madurska. A set-by-set analysis method for predicting the outcome of professional singles tennis matches. MEng computing-final year project, Imperial College London, amm208@doc.ic.ac.uk, June 2012.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *ournal of Machine Learning Research*, 12:2825–2830, 2011.

[8] Jeff Sackmann. Atp tennis rankings, results, and stats, 2017.

[9] Michal Sipko. Machine learning for the prediction of professional tennis matches. MEng computing - final year project, Imperial College London, June 2015.